Express Mail Label No. EL138268973US

06/03/98

UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No. INTL-0033-US (P5395)

Total Pages in this Submission 24

TO THE ASSISTANT COMMISSIONER FOR PATENTS
Box Patent Application
Washington, D.C. 20231

Transmitted herewith for filing under 35 U.S.C. 111(a) and 37 C.F.R. 1.53(b) is a new utility patent application for an invention entitled:

BINARY COMPATIBLE SOFTWARE OBJECTS

and invented by:

Kenneth S. Knapton, III

If a CONTINUATION APPLICATION, *check appropriate box and supply the requisite information:*

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

☐ **Continuation** ☐ **Divisional** ☐ **Continuation-in-part (CIP)** of prior application No.: _____

Enclosed are:

Application Elements

1. ☒ Filing fee as calculated and transmitted as described below

2. ☒ Specification having 11 pages and including the following:

   a. ☒ Descriptive Title of the Invention

   b. ☐ Cross References to Related Applications *(if applicable)*

   c. ☐ Statement Regarding Federally-sponsored Research/Development *(if applicable)*

   d. ☐ Reference to Microfiche Appendix *(if applicable)*

   e. ☒ Background of the Invention

   f. ☒ Brief Summary of the Invention

   g. ☒ Brief Description of the Drawings *(if drawings filed)*

   h. ☒ Detailed Description

   i. ☒ Claim(s) as Classified Below

   j. ☒ Abstract of the Disclosure

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.
INTL-0033-US (P5395)
Total Pages in this Submission
24

Application Elements (Continued)

3. ☒ Drawing(s) *(when necessary as prescribed by 35 USC 113)*
   a. ☒ Formal Number of Sheets 3
   b. ☐ Informal Number of Sheets

4. ☒ Oath or Declaration
   a. ☒ Newly executed *(original or copy)* ☐ Unexecuted
   b. ☐ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional application only)*
   c. ☒ With Power of Attorney ☐ Without Power of Attorney
   d. ☐ *DELETION OF INVENTOR(S)*
   Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. 1.63(d)(2) and 1.33(b).

5. ☐ Incorporation By Reference *(usable if Box 4b is checked)*
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. ☐ Computer Program in Microfiche *(Appendix)*

7. ☐ Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all must be included)*
   a. ☐ Paper Copy
   b. ☐ Computer Readable Copy *(identical to computer copy)*
   c. ☐ Statement Verifying Identical Paper and Computer Readable Copy

Accompanying Application Parts

8. ☒ Assignment Papers *(cover sheet & document(s))*

9. ☐ 37 CFR 3.73(B) Statement *(when there is an assignee)*

10. ☐ English Translation Document *(if applicable)*

11. ☐ Information Disclosure Statement/PTO-1449 ☐ Copies of IDS Citations

12. ☐ Preliminary Amendment

13. ☒ Acknowledgment postcard

14. ☒ Certificate of Mailing

☐ First Class ☒ Express Mail *(Specify Label No.):* EL138268973US

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.
INTL-0033-US (P5395)
Total Pages in this Submission
24

Accompanying Application Parts (Continued)

15. ☐ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*

16. ☐ Additional Enclosures *(please identify below):*

Fee Calculation and Transmittal

CLAIMS AS FILED

For	#Filed	#Allowed	#Extra		Rate	Fee
Total Claims	17	- 20 =	0	x	$22.00	$0.00
Indep. Claims	4	- 3 =	1	x	$82.00	$82.00
Multiple Dependent Claims (check if applicable) ☐						$0.00
					BASIC FEE	$790.00
OTHER FEE *(specify purpose)*						$0.00
					TOTAL FILING FEE	$872.00

☒ A check in the amount of $872.00 to cover the filing fee is enclosed.

☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. 02-1504(IN33) as described below. A duplicate copy of this sheet is enclosed.

- ☐ Charge the amount of as filing fee.
- ☒ Credit any overpayment.
- ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.
- ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

Signature

Dated: June 3, 1998

Timonthy N. Trop, Reg. No. 28,994
Trop, Pruner, Hu & Miles, P.C.
8550 Katy Freeway, Suite 128
Houston, Texas 77024
(713) 468-8880
(713) 468-8883 Fax

cc:

PATENT
ATTORNEY DOCKET NO. INTL0033 (P5395)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **BINARY COMPATIBLE SOFTWARE OBJECTS**

APPLICANT: **KENNETH S. KNAPTON, III**

Express Mail No. EL138268973US
Date Mailed: June 3, 1998

Background

This invention relates generally to object oriented software technologies and particularly to software objects which are binary compatible.

ActiveX controls are a subset of the COM object oriented software technology. COM can use a variety of different object oriented program languages such as C++, Java and Visual Basic. ActiveX controls are typically plugged into a control container which is a type of client.

The ActiveX controls self-register on a computer in a database. In Windows®-based platforms, the database is called a registry. The registry provides a way for a control to advise the client about the control's functionality. More specifically, the ActiveX control places keys in the registry or database to let the container know its functionality. The registry includes information which identifies a particular control or object including Globally Unique Identifiers (GUIDs), Category Identifiers (CATIDs), and Class Identifiers (CLSIDs).

A layer class, wrapper or interface definition is a source code level version of a COM object. It provides an interface between the container or client and the object which may be an ActiveX control. Additional controls may be inserted, snapped in or "plugged in" to a container that already has one or more controls. A plug-in control is source compatible if a new version of the control works unchanged in a container application but the user program must be rebuilt. That is, the application program must be recompiled and then the application can be run without further change.

With a "binary compatible" control, a new version

can be plugged into an existing application that was designed and built for the old version. However, the conventional wisdom in the field is that the plug-in must appear to the container as if it were the old version in order for the plug-in to be binary compatible. That is, the plug-in must support the old CLSID and all interfaces exactly as they were (that is, with the same IIDs, names, dispids, parameters and so forth). See Denning, Adam, "ActiveX Controls Inside Out", Microsoft Press (1997), p. 131. Thus, the conventional wisdom holds that in order to be binary compatible, the same identifiers and interfaces must be used for the plug-in.

A GUID is conventionally hard coded into a layer class. Other objects can then be used with a given container; however, they must have the same interface and GUID in order to work with the layer class in a binary compatible fashion.

Thus, there is a continuing need to enable objects, with different interfaces and/or different GUIDs, to snap in to a container or client environment.

Summary

In accordance with one aspect of the present invention, a method for object oriented programming includes creating a first object, having a first identifier associated with a first client. Thereafter, a second object having a second identifier is inserted, such that the second object is associated with the first client. Even though the first and second identifiers are different, the second object is used with the first client, without recompiling.

Brief Description of the Drawings

Fig. 1 is a high level flow diagram in accordance with one aspect of the present invention;

Fig. 2 is a more detailed flow chart for implementing a flow in accordance with Fig. 1; and

Fig. 3 is a conceptual depiction of an approach to object oriented programming.

Detailed Description

Referring to Fig. 1, a method for object oriented programming may be implemented in any object oriented programming technology including COM, ActiveX, Java, Visual Basic, C++ and the like. As indicated in block 10, a first object with a first GUID is registered on the system. Then a second object with a second GUID may be registered as indicated in block 12. Time may pass between the first and second registrations. The registration may be done in any conventional database for registering objects including the so-called registry utilized in Windows®-based platforms.

As indicated in block 14, it is thereafter possible to selectively access one of the first and second objects using the same client or container despite the fact that the objects have different GUIDs. In addition, these first and second objects may be accessed using the same container or client despite the fact that they have different interfaces, for example, in connection with COM based objects. Of course, the dispids and parameters for the two objects still would be the same. However, the first object or the second object may be selectively accessed after snapping in the second object.

In this way, a truly binary compatible object system

may be developed in which a second object may be snapped into a container or client containing a first object and the second object may be utilized without recompiling. No recompiling is necessary even though the first and second objects have different GUIDs and even if the first and second objects have different interfaces, for example, in the case of COM applications.

While these techniques are applicable to a variety of technologies, they are particularly applicable to ActiveX controls wherein first and second controls may be snapped into a container. In this way the container can selectively access either the first or the second object without recompiling. In applications using a layer class or wrapper, the layer class may then be associated with more than one object. A new ActiveX control can be snapped in without creating a new layer class each time, which would require recompiling.

Referring now to Fig. 2, a more detailed flow chart for implementing the technology of Fig. 1 in COM technology begins with block 16. At block 16, each object for a given client or container is registered. At block 18, a layer class is appropriately programmed (using the SetGUID and GetGUID methods described below) to enable the GUIDs for any of the registered objects to be selected. Thereafter, it is then possible to access either object despite the fact that the second object that is snapped in may have a different GUID than the first object. In connection with applications involving ActiveX controls using dynamic linked libraries or DLLs, the registration is normally accomplished using the DllRegisterServer() method in Windows®-based applications. In non-ActiveX applications when DLLs are not used, the

registration may be done with EXEs. In this way, the GUIDs of the respective objects are registered in a database (or in Windows® applications, at the registry).

SetGUID and GetGUID methods are implemented in an application involving a layer class. GetGUID, which is part of the layer class, provides the GUID that is set when the layer class calls for its own GUID. The SetGUID method, which is part of the layer class, sets the GUID in the layer class for the desired object, as shown at block 22. Thus, SetGUID sets the new GUID in the layer class and GetGUID returns the GUID from the database.

In alternative embodiments it is possible to use the CoCreateInstance method to send in the GUID of the second object when that object is referenced. The GUID of the second object is sent in for the snapped in second object and the same interface definition may be utilized for both objects. In this case, the SetGUID and the GetGUID method are unnecessary and manipulation of the layer class is likewise unnecessary.

Using either technique, it is possible to dynamically create new functions in the future for a given client or container. This can be done in a truly binary compatible fashion without requiring identifiers to be identical or interfaces to be identical, or recompiling.

Referring again to Fig. 2, one can access the first object, as indicated in block 24, thereafter obtain the GUID for a second object, as indicated in block 26, and set the GUID for the second object in place of the GUID for the first object, as indicated in block 28. As indicated in block 30, the second object may be accessed by the same container or client that previously would have accessed the

first object. This is done in a binary compatible fashion without recompiling.

These techniques are applicable, not only to in process servers such as DLLs, but also to remote applications, such as Distributed COMs (DCOMs). Generally the techniques described herein are applicable to any object which can be snapped into a container or client situation.

Referring now to Fig. 3, a pair of objects 32 and 34 have different GUIDs and different interfaces. The interface for the object 32 is indicated as I1 and the GUID for object 32 is indicated as GUIDA. Similarly, the GUID for object 34 is GUIDB and the interface for object 34 is I2.

The compiled application, indicated within the dotted line box 36, includes, in the illustrated embodiment, a layer class or wrapper 38 and the client or container 46. As indicated, the layer class communicates with the client 46.

The layer class 38 includes the SetGUID method 42 and the GetGUID method 40. The layer class 38 also stores the selected GUID 44. In the illustrated embodiment, the selected GUID is the GUIDA as indicated by the arrow between the layer class and the object 32. However, the layer class can obtain the GUID for the object 34 through the client from the system database 48. The system database or registry includes GUIDs 50 and 52 in the illustrated embodiment. Thus, the client can obtain GUIDs for desired objects from the system database 48 and the layer class can access the corresponding object without requiring recompiling.

In prior systems, the objects 32 and 34 would need

to have the same GUIDs and interfaces to enable a binarily compatible snap-in system. Now snap-in objects with different GUIDs and different interfaces may be used to implement new functions as necessary in the future without recompiling. In this way, a layer class may be created that has selectively programmable GUIDs for more than one object.

With embodiments of the present invention, one can extend a current technology by allowing updates and new functions. That is, a new object may be used in place of its predecessor. The new object may also be used selectively in conjunction with its predecessor.

While the present invention has been described with respect to a limited number of preferred embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. For example, while the illustrated embodiments illustrate a COM or ActiveX controls application, those skilled in the art will appreciate that similar approaches can be used in other programming technologies and languages. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for object oriented programming comprising: creating a first object having a first identifier, said object associated with a first client;

inserting a second object having a second identifier, said second object associated with the first client, said first and second identifiers being different; and

using said second object with said first client in place of the first object without recompiling

2. The method of claim 1 including creating a first COM object having a first globally unique identifier, said first COM object associated with a first container, inserting a second COM object having a second globally unique identifier, said second COM object associated with the first container, the first and second globally unique identifiers being different, and using said second COM object with the first container without recompiling.

3. The method of claim 2 including providing a layer class and setting said globally unique identifier in said layer class.

4. The method of claim 1 including creating a layer class that interfaces with one of a plurality of globally unique identifiers of objects associated with said layer class.

5. The method of claim 1 including using said first object again with said first client in place of said

second object without recompiling.

6. A method for object oriented programming comprising:

registering a first object with a first globally unique identifier;

registering a second object with a second globally unique identifier; and

selectively accessing one of said first and second objects without recompiling.

7. The method of claim 6 including creating a source code version of said objects, and programming said globally unique identifiers into a layer class.

8. The method of claim 7 including getting the globally unique identifier for each object from a database and setting each globally unique identifier in said layer class.

9. A container for a software object comprising:

one or more objects, said container adapted to selectively work with first and second objects having different identifiers.

10. The container of claim 9 including a layer class adapted to selectively utilize the identifier of either said first or second object.

11. The container of claim 10 wherein said layer class includes a first function that obtains globally unique

identifiers from a system database and a second function that sets globally unique identifiers in the layer class.

12. A computer readable storage medium for storing a program including instructions for causing a computer to:

create an object having a first identifier, said object associated with a first client;

insert a second object having a second identifier, said second object associated with the first client, said first and second identifiers being different; and

use said second object with said first client in place of said first object without recompiling.

13. The medium of claim 12 wherein said objects are COM objects.

14. The medium of claim 13 wherein said COM objects are ActiveX controls.

15. The medium of claim 13 wherein said identifiers are globally unique identifiers.

16. The medium of claim 15 including one or more instructions for storing a program instructions for causing a computer to create a layer class having selectively programmable globally unique identifiers for more than one object.

17. The medium of claim 16 including instructions for causing a computer to obtain globally unique identifiers and setting the identifiers in the layer class.

BINARY COMPATIBLE SOFTWARE OBJECTS

Abstract

An object oriented programming technology enables multiple objects to be snapped into a given container or client. Thus, for example, multiple ActiveX controls can be snapped into one given container and these controls may be accessed in a fashion which can be truly termed binary compatible. That is, each of the controls can be accessed without requiring recompiling even when they have different identifiers and/or different interfaces. In one embodiment, this may be accomplished by providing SetGUID and GetGUID functions in a layer class associated with the objects. The identifier of the desired object can be obtained from the system database and set in the layer class to selectively access one of at least two objects in the same container without recompiling.

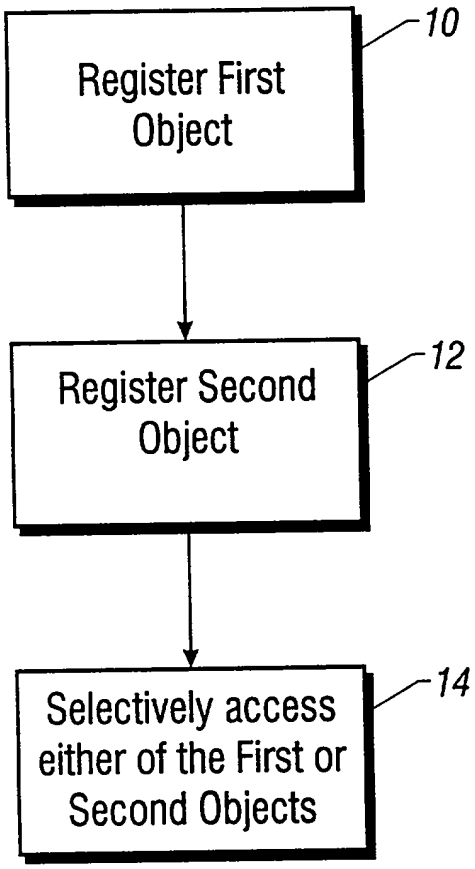


Figure 1

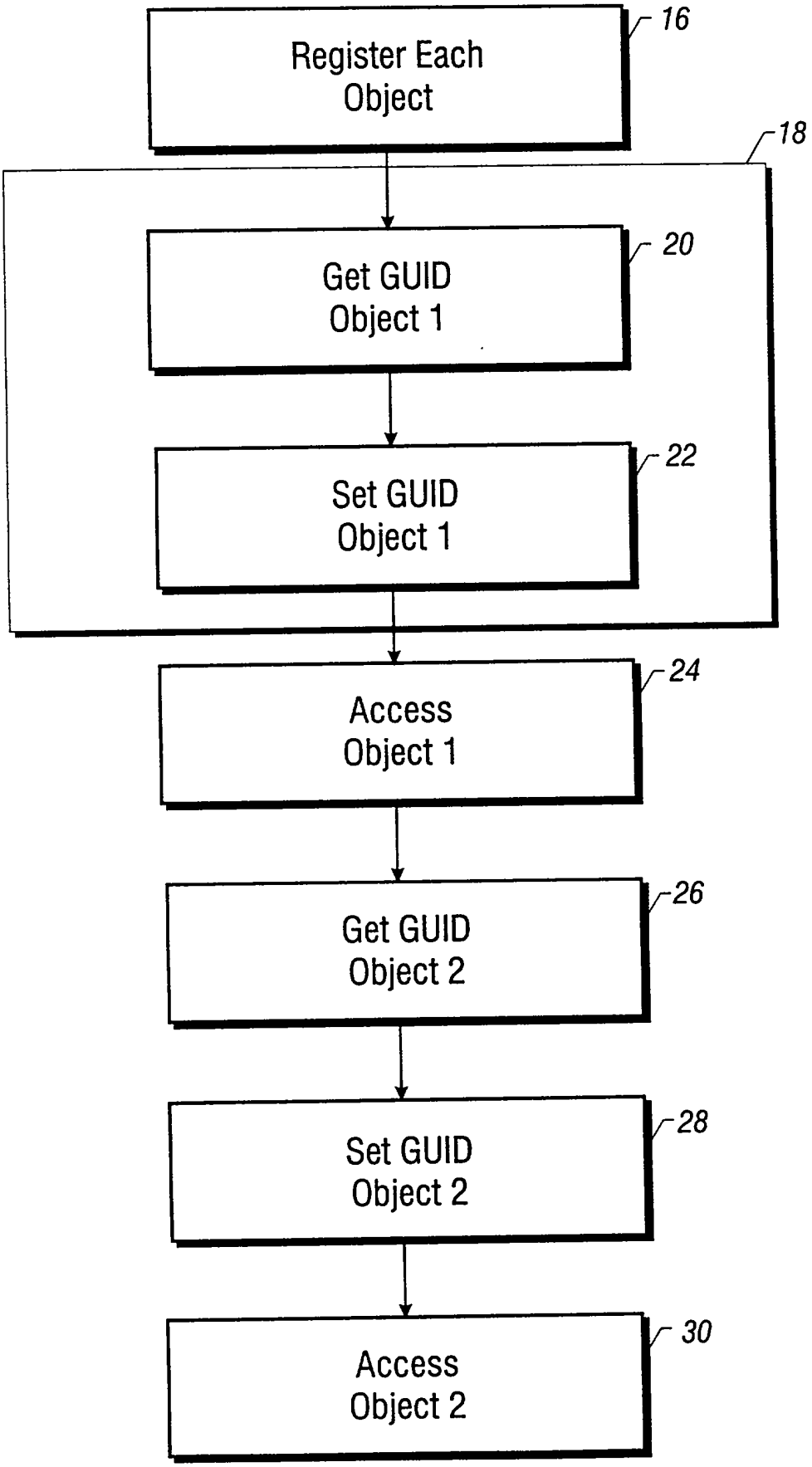


Figure 2

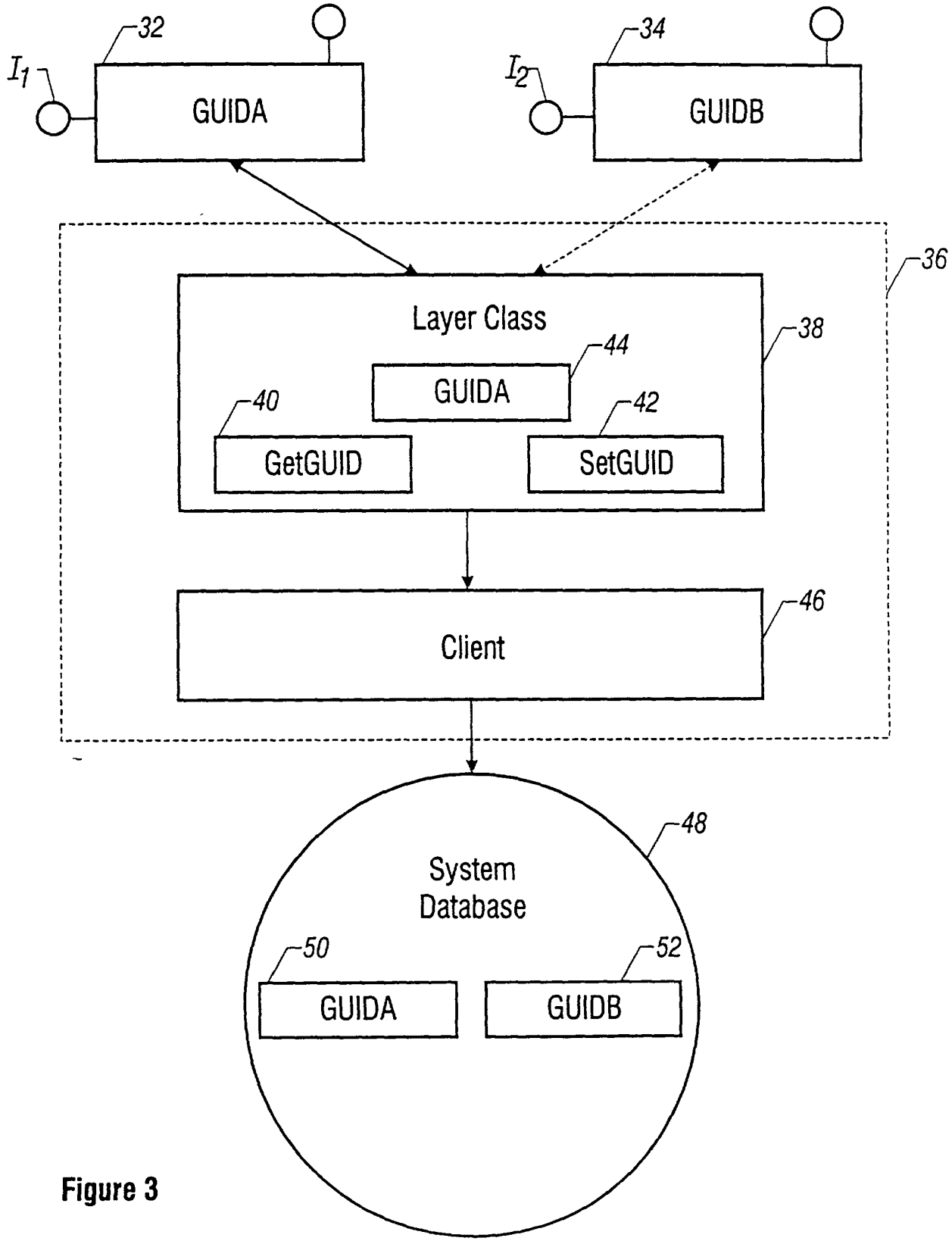

32
34
I_1
I_2
GUIDA
GUIDB
36
Layer Class
38
44
GUIDA
40
42
GetGUID
SetGUID
46
Client
48
System
Database
50
52
GUIDA
GUIDB

Figure 3

Attorney's Docket No.: INTL-0033-US (P5395) PATENT

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION (FOR INTEL CORPORATION PATENT APPLICATIONS)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

BINARY COMPATIBLE SOFTWARE OBJECTS

the specification of which

X	is attached hereto.
	was filed on _____ As
	United States Application Number _____
	or PCT International Application Number _____
	and was amended on _____
	(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate Issued before the date of this application in any country foreign to the United States of American on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):			Priority Claimed	
Number	(Country)	(Day/Month/Year Filed)	Yes	No
Number	(Country)	(Day/Month/Year Filed)	Yes	No
Number	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under title 35, United States Code, Section 119(e) of the United States provisional application(s) listed below:

(Application Number)	(Filing Date)
(Application Number)	(Filing Date)

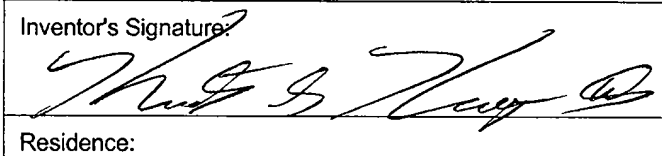
I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

(Application Number)	Filing Date	(Status-patented, pending, abandoned)
(Application Number)	Filing Date	(Status-patented, pending, abandoned)

I hereby appoint Timothy N. Trop, Reg. No. 28,994; Fred G. Pruner, Jr., Reg. No. 40,779, Dan C. Hu, 40,025; Coe F. Miles, Reg. No. 38,559, my patent attorneys, of TROP, PRUNER, HU & MILES, P.C., with offices located at 8550 Katy Freeway, Ste. 128, Houston, TX 77024, telephone (713) 468-8880, and Joseph R. Bond, Reg. No. 36,458; Richard C. Calderwood, Reg. No. 35,468; Sean Fitzgerald, Reg. No. 32,027; David J. Kaplan, Reg. No. 41,105; Leo V. Novakoski, Reg. No. 37,198; Naomi Obinata, Reg. No. 39,320; Thomas C. Reynolds, Reg. No. 32,488; Steven P. Skabrat, Reg. No. 36,279; Howard A. Skaist, Reg. No. 36,008; Steven C. Stewart, Reg. No. 33,555; Raymond J. Werner, Reg. No. 34,752; and Charles K. Young, Reg. No. 39,425; my patent attorneys, of INTEL CORPORATION; with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Timothy N. Trop, TROP, PRUNER, HU & MILES, P.C., 8550 Katy Freeway, Ste. 128, Houston, TX 77024 and direct telephone calls to Timothy N. Trop, (713) 468-8880.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: **Kenneth S. Knapton, III**	
Inventor's Signature: 	Date: 6-1-98
Residence: **1239 N 200 W, Orem, UT 84057**	Citizenship: **USA**
Post Office Address: **1239 N 200 W, Orem, UT 84057**	

INTL-0039-US